

University of Groningen

Solutions of equations in languages

Hesselink, Wim H.

Published in:
Formal Aspects of Computing

DOI:
[10.1007/s00165-009-0123-x](https://doi.org/10.1007/s00165-009-0123-x)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2010

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):
Hesselink, W. H. (2010). Solutions of equations in languages. *Formal Aspects of Computing*, 22(5), 537-545. <https://doi.org/10.1007/s00165-009-0123-x>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Solutions of equations in languages

Wim H. Hesselink

Department of Computing Science, University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands.
E-mail: w.h.hesselink@rug.nl

Abstract. A context-free grammar corresponds to a system of equations in languages. The language generated by the grammar is the smallest solution of the system. We give a necessary and sufficient condition for an arbitrary solution to be the smallest one. We revive an old criterion to decide that a grammar has a unique solution. All this fits in an approach to search for a grammar for an arbitrary language that is given by other means. The approach is illustrated by the derivation of a grammar for a certain set of bit strings. The approach is used to give an elegant derivation of the grammar for a language accepted by a pushdown automaton.

Keywords: Language, Grammar, Context-free, Pushdown automaton, Unguarded recursion

1. Introduction

Ever since their introduction by Chomsky in 1956, (context-free) grammars have been used to generate (context-free) languages. Alternatively, however, a grammar can be used to define a system of equations in languages. The language generated by the grammar is the smallest solution of the system. This is well-known, e.g. [MG87], but it is rarely mentioned in courses on languages. It comes up in exercises and investigations where one wants to determine a grammar for a language that is defined by other means. A disturbing point is of course that the system of equations may have more than one solution.

In Section 2, we show that a solution of a system of equations corresponding to a grammar is the smallest solution if and only if it has a so-called filtration. The idea is not surprising. What is surprising is that we could not find it in the literature. This result implies the well-known fact that the language generated by the grammar is the smallest solution. We proceed to prove that, if a grammar has more than one solution, it has a nonempty “unguarded set”. This criterion was new to us, but a referee found it in two papers from 1965. Section 3 is an aside about fixed point equations, which also contains the algorithm to determine the greatest unguarded set.

The results of Sections 2 and 3 are applied in Sections 4 and 5 to propose and illustrate a method for the determination of a grammar for a language that is defined by other means. In Section 4, we introduce the method, and apply it to the language of the bit strings such that the number of 0s preceded by an even number of 1s equals the number of 0s preceded by an odd number of 1s.

In Section 5, we use the method to prove the classical result that the language accepted by a pushdown automaton is context-free. In comparison with the classical treatments of, e.g., [Cho62, HMU01, Sud06], we simplify the resulting grammar and straighten the proof. Conclusions are drawn in Section 6.

2. Solutions of a grammar

Let $G = (V, \Sigma, P, S)$ be a context-free grammar. Recall that this means that V is a finite set of nonterminal symbols, Σ is a finite set of terminal symbols (V and Σ are disjoint), P is a finite set of production rules $X \rightarrow x$ with $X \in V$ and $x \in B^*$ where $B = V \cup \Sigma$, and $S \in V$ is the start symbol. The corresponding derivation relation \Rightarrow_G on B^* is defined by

$$u \Rightarrow_G v \quad \equiv \quad \exists y, z \in B^*, (X \rightarrow x) \in P : u = yXz \wedge v = yxz.$$

Relation \Rightarrow_G^* is the reflexive transitive closure of \Rightarrow_G . The language generated by the grammar is $\text{Gen}(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$, where S is the start symbol of grammar G . More generally, we define $H(X) = \{w \in \Sigma^* \mid X \Rightarrow_G^* w\}$ for every nonterminal $X \in V$.

Given a set-valued function $F : V \rightarrow \mathbb{P}(\Sigma^*)$, let $F^* : B^* \rightarrow \mathbb{P}(\Sigma^*)$ be defined inductively by

$$\begin{aligned} F^*(\varepsilon) &= \{\varepsilon\}, \\ F^*(au) &= \{a\}F^*(u) \quad \text{for } a \in \Sigma, u \in B^*, \\ F^*(Xu) &= F(X)F^*(u) \quad \text{for } X \in V, u \in B^*, \end{aligned}$$

where ε is the empty string, and concatenation of languages is defined by $L_0L_1 = \{uv \mid u \in L_0, v \in L_1\}$ in the usual way.

We now apply this to function H introduced above. By induction over the length of u , one can prove that

$$(0) \quad H^*(u) = \{w \in \Sigma^* \mid u \Rightarrow_G^* w\} \quad \text{for all } u \in B^*.$$

For any $X \in V$ and $w \in \Sigma^*$, we have

$$X \Rightarrow_G^* w \quad \equiv \quad \exists x : (X \rightarrow x) \in P \wedge x \Rightarrow_G^* w.$$

By (0), this implies that

$$H(X) = \bigcup \{H^*(x) \mid x : (X \rightarrow x) \in P\} \quad \text{for all } X \in V.$$

The starting point of this paper is the question whether function H is characterized uniquely by this equation.

We therefore define a function $F : V \rightarrow \mathbb{P}(\Sigma^*)$ to be a *solution* of grammar G if it satisfies the condition:

$$(1) \quad F(X) = \bigcup \{F^*(x) \mid x : (X \rightarrow x) \in P\} \quad \text{for all } X \in V.$$

We thus have that H is a solution of grammar G . In fact, it is well-known that H is the smallest solution of G , see [MG87, Thm. 1.21]. To prove this and put it in context, we develop a little theory that can be reused later on.

Let F be a solution of grammar G . A *filtration* of F is an infinite family of functions $F_n : V \rightarrow \mathbb{P}(\Sigma^*)$, $n \in \mathbb{N}$, with for all $X \in V$ the properties

$$\begin{aligned} F(X) &= \bigcup_n F_n(X), \\ F_0(X) &= \emptyset, \\ F_{n+1}(X) &\subseteq \bigcup \{F_n^*(x) \mid x : (X \rightarrow x) \in P\} \quad \text{for all } n \in \mathbb{N}. \end{aligned}$$

For filtrations in mathematics, one usually also requires that $F_n(X) \subseteq F_{n+1}(X)$ for all n and X . This is not needed for the present purposes. A solution that has a filtration is called a *filtered solution*.

Theorem 1 *Let F be a filtered solution of grammar G . Then F is the smallest solution of G with respect to subset inclusion.*

Proof. Let F' be a second solution of G . It suffices to prove that $F(X) \subseteq F'(X)$ for all X . Let $(F_n \mid n \in \mathbb{N})$ be a filtration of F . By the first clause above, it suffices to prove that $F_n(X) \subseteq F'(X)$ for all X and n . This is proved by induction over n . For $n = 0$, it follows from the second clause. If $F_n(X) \subseteq F'(X)$ for all X , then $F_n^*(u) \subseteq F'^*(u)$ for all $u \in B^*$, and hence $F_{n+1}(X) \subseteq F'(X)$ for all X by the third clause and formula (1) for F' . \square

As announced, part of the next result is classical:

Theorem 2 *Function H is a filtered solution of grammar G , and therefore the smallest solution of grammar G with respect to subset inclusion. Every filtered solution of G equals H .*

Proof. By Theorem 1, it remains to prove that solution H has a filtration. We let $H_n(X)$ consist of the terminal strings w with $X \Rightarrow_G^k w$ for some $k \leq n$, where \Rightarrow_G^k is the relational composition of k copies of \Rightarrow_G . This clearly defines a filtration of H . \square

Corollary 3 *A solution of a grammar is the smallest one with respect to subset inclusion if and only if it is filtered.*

Example. Let $a \in \Sigma$. Consider the grammar G over Σ given by the production rules $S \rightarrow a \mid SSS$. Here $V = \{S\}$ and P holds the rules $S \rightarrow a$ and $S \rightarrow SSS$. The language $H(S)$ consists of the strings a^n with n odd. Yet, grammar G also has the solution F with $F(S) = \Sigma^*$ (and many others if $\Sigma \neq \{a\}$).

This example suggests that the occurrence of multiple solutions is due to recursive productions without terminal symbols. This is confirmed by the next result. We use the notation $|w|$ for the length of a string w .

Let $N \subseteq V$ be the set of *nullables* of G , i.e., $N = \{X \in V \mid X \Rightarrow_G^* \varepsilon\}$. Let a set of nonterminals $T \subseteq V$ be called *unguarded* if every $X \in T$ has a production $(X \rightarrow x) \in P$ with $x \in (N \cup T)^* T (N \cup T)^*$.

Theorem 4 *Let G have a solution $F \neq H$. Then G has a nonempty unguarded set T of nonterminals.*

Proof. We have $H(X) \subseteq F(X)$ for every $X \in V$ by Theorem 2. As $F \neq H$, we can choose a string u of minimal length for which the set of “spurious producers” $T = \{X \in V \mid u \in F(X) \setminus H(X)\}$ is nonempty. We prove that T is unguarded.

Let $X \in T$. Because F is a solution and $u \in F(X)$, there is a production $(X \rightarrow x) \in P$ with $u \in F^*(x)$. Write $x = x_1 \cdots x_k$ with $x_i \in B$. Then there are v_1, \dots, v_k with $u = v_1 \cdots v_k$ and $v_i \in F(x_i)$ or $v_i = x_i \in \Sigma$. If $v_i \in H(x_i)$ or $v_i = x_i \in \Sigma$, for all indices i , then $u \in H(x)$ as is not the case. Therefore, $k \geq 1$ and there is at least one index j with $x_j \in V$ and $v_j \notin H(x_j)$. By minimality of $|u|$, we have $|u| \leq |v_j|$. It follows that $v_j = u$ and that $v_i = \varepsilon$ for all $i \neq j$. In particular, $x_j \in T$. For $i \neq j$, we have $x_i \in V$ and $\varepsilon \in F(x_i)$. Clearly, $\varepsilon \in H(x_i)$ implies $x_i \in N$. If $\varepsilon \notin H(x_i)$, then $u = \varepsilon$ by minimality of $|u|$, implying that $x_i \in T$. In either case, we have $x_i \in N \cup T$. This proves $x \in (N \cup T)^* T (N \cup T)^*$. Therefore T is unguarded. \square

As noted by a referee, Theorem 4 is equivalent to Theorem 6 of [Red65] and probably also to the Corollary of Theorem 16 in [Bod65a, Bod65b].

Conversely, the existence of a nonempty unguarded set is not sufficient for the existence of multiple solutions because “real productions can hide the spurious ones”. For example, the grammar $S \rightarrow \varepsilon \mid a \mid SS$ over the terminal alphabet $\Sigma = \{a\}$ has the nonempty unguarded set $\{S\}$, but it has a unique solution: Σ^* .

Theorem 4 has the following corollary, which seems to be well-known (compare [McW71, Theorem 1]):

Corollary 5 *Assume that every nonempty production of grammar G contains at least one terminal symbol. Then H is the only solution of G .*

Proof. Let $T \subseteq V$ be unguarded. If $X \in T$, then X has a production $(X \rightarrow x) \in P$ with $x \in (N \cup T)^* T (N \cup T)^* \subseteq V^+$, contradicting the assumption on G . Therefore T is empty. Theorem 4, therefore, implies that H is the only solution. \square

Remark. Theorem 7 of [Kup97] shows that disallowing the empty string ε makes the situation much simpler: let G be a grammar, in which every $X \in V$ has at least one production, and that has no ε productions or unit productions (i.e., productions $X \rightarrow Y$ with $X, Y \in V$); then G has a unique solution in the set of languages that do not contain ε .

3. Various fixed points

In order to apply Theorem 4, we need to know whether a grammar has a nonempty unguarded set. We therefore present an algorithm to determine the greatest unguarded set. This algorithm consists of two steps: first determine the set of the nullables N , and then the greatest unguarded set T .

The algorithm to determine the set N of the nullables is well known, e.g., [Sud06, Section 4.2]. It uses a set transformer $\text{pinv} : \mathbb{P}(V) \rightarrow \mathbb{P}(V)$ defined by, for $U \subseteq V$:

$$\text{pinv}(U) = \{X \in V \mid \exists x : (X \rightarrow x) \in P \wedge x \in U^*\}.$$

As $\emptyset^* = \{\varepsilon\}$, we have that X is nullable if and only if $X \in \text{pinv}^n(\emptyset)$ for some $n \in \mathbb{N}$. The set of the nullables is therefore computed by

```
var  $N := \emptyset$ ,  $U := \text{pinv}(N)$ ;
while  $N \neq U$  do  $N := U$ ;  $U := \text{pinv}(N)$  end.
```

This repetition terminates because N grows in every step and the set V is finite. In the postcondition, N is the set of the nullables and also the smallest fixed point of pinv .

The algorithm to determine the greatest unguarded set is strikingly similar. It uses a set transformer $\text{ung} : \mathbb{P}(V) \rightarrow \mathbb{P}(V)$ defined by, for $T \subseteq V$:

$$\text{ung}(T) = \{X \in T \mid \exists x : (X \rightarrow x) \in P \wedge x \in (N \cup T)^* T (N \cup T)^*\}.$$

Clearly, $T \subseteq V$ is unguarded if and only if T is a fixed point of ung , i.e., $T = \text{ung}(T)$. The greatest fixed point of ung , and hence the greatest unguarded set, can be obtained by repeated application of ung starting with V :

```
var  $T := V$ ,  $U := \text{ung}(T)$ ;
while  $T \neq U$  do  $T := U$ ;  $U := \text{ung}(T)$  end.
```

This repetition has the invariant that T contains every unguarded set and that $U = \text{ung}(T)$. It has therefore the postcondition that T is the greatest unguarded set. It terminates because T is a finite set that shrinks in every step. The greatest unguarded set is called the characteristic set in [Red65].

Example. Let G be the grammar over $\Sigma = \{a, b, c\}$ given by

```
 $S \rightarrow \varepsilon \mid AS$ 
 $A \rightarrow a \mid BB$ 
 $B \rightarrow b \mid cS$ .
```

In this case, the set N of nullables equals $\{S\}$. We have $\text{ung}(V) = \{S, A\}$, $\text{ung}^2(V) = \{S\}$, and $\text{ung}^3(V) = \emptyset$. The greatest unguarded set is therefore empty. Theorem 4 therefore implies that grammar G has only one solution.

Whereas N and T are thus smallest and greatest fixed points of monotonic functions from $\mathbb{P}(V)$ to itself, equation (1) is a fixed point equation of the form $F = g(F)$ for the function g from the set $V \rightarrow \mathbb{P}(\Sigma^*)$ to itself given by

$$g(F)(X) = \bigcup \{F^*(x) \mid x : (X \rightarrow x) \in P\} \quad \text{for all } X \in V.$$

In other words, solutions of grammar G are just fixed points of function g . If one gives the set $V \rightarrow \mathbb{P}(\Sigma^*)$ the order of pointwise inclusion, then it is a complete lattice and function g is monotonic. Therefore, the Theorem of Knaster–Tarski is applicable and the set of fixed points of g is a complete lattice. In particular, g has a smallest fixed point, conventionally denoted by μg . Theorem 2 says that $\mu g = H$. As noted by a referee, Theorem 1 can easily be generalized to a monotonic function g from an abstract lattice to itself. This paper can therefore be regarded as an application or extension of fixed point theory [Kup97]. The greater context, however, is not needed, nor does it help much.

4. A grammar for a specific language

When a language is not given by how its strings are generated, but by some other means, it is not directly clear how to derive a grammar for it. We propose the following approach. It is a method that consists of three steps: first derive a finite set of equations for the language, then recognize in this set of equations an instance of formula (1) for a specific grammar, and finally argue that the language corresponds to the smallest solution. The method can be applied fruitfully, e.g., to Exercise 5.1.1(d) of [HMU01], which asks for a context-free grammar for the set of all bit strings with twice as many 0s as 1s.

Here we treat the related, but more challenging problem to determine a grammar for the set L of the bit strings w such that the number of 0s preceded by an even number of 1s equals the number of 0s preceded by an odd number of 1s.

The terminal alphabet is $\Sigma = \{0, 1\}$. Let $f : \Sigma^* \rightarrow \mathbb{Z}$ be the function that gives the difference between the two counts: $f(w)$ is the number of 0s in w preceded by an even number of 1s minus the number of 0s in w preceded by an odd number of 1s. It is clear that $L = \{w \in \Sigma^* \mid f(w) = 0\}$, and that function f satisfies the recursive equations $f(\varepsilon) = 0$, $f(0w) = 1 + f(w)$, and $f(1w) = -f(w)$.

In order to get L determined by a finite set of equations in languages, we introduce a signature function $s : \Sigma^* \rightarrow \mathbb{Z}$ that gives ± 1 according to whether the number of 1s in w is even or odd. So, s satisfies $s(\varepsilon) = 1$, $s(0w) = s(w)$, and $s(1w) = -s(w)$. It is now easy to prove by induction that function s is multiplicative in the sense that $s(xy) = s(x) \cdot s(y)$. By induction on x , we prove for f the rule

$$(2) \quad f(xy) = f(x) + s(x) \cdot f(y).$$

Indeed, it holds for $x = \varepsilon$ because $f(\varepsilon) = 0$ and $s(\varepsilon) = 1$, and, if it holds for x , then it holds for $0x$ because of $f(0xy) = 1 + f(xy) = 1 + f(x) + s(x) \cdot f(y) = f(0x) + s(0x) \cdot f(y)$, and for $1x$ because of $f(1xy) = -f(xy) = -f(x) - s(x) \cdot f(y) = f(1x) + s(1x) \cdot f(y)$.

We now define the languages $K_{i,j} = \{w \mid f(w) = i \wedge s(w) = j\}$. We have

$$(3) \quad L = K_{0,1} \cup K_{0,-1}.$$

The language $K_{i,j}$ contains ε if and only if $i = 0$ and $j = 1$. Every nonempty string in $K_{i,j}$ starts with 0 or 1. The recursive definitions of functions f and s therefore imply that

$$(4) \quad K_{i,j} = \{\varepsilon \mid i = 0 \wedge j = 1\} \cup \{0\}K_{i-1,j} \cup \{1\}K_{-i,-j}.$$

The language L is completely determined by the equations (3) and (4), but then we need infinitely many equations and, hence, a grammar with infinitely many nonterminals. This implies that we need a deeper analysis.

Consider $w \in \Sigma^*$ with $f(w) < 0$. Then w has a smallest prefix u with $f(u) < 0$. Since u is minimal and $f(u) < 0$, we can write $u = xa$ with $f(x) \geq 0$ and $a \in \Sigma$. Formula (2) gives $f(u) = f(x) + s(x)f(a)$ and hence $f(a) \neq 0$ because $f(u) < 0 \leq f(x)$. Therefore $a = 0$ and $f(a) = 1$ and $s(x) = -1$ and $f(x) = 0$. This implies $w = x0y$ for some $x \in K_{0,-1}$ and some string y .

In general, if $w = x0y$ with $x \in K_{0,-1}$, then $f(w) = -f(0y) = -1 - f(y)$ and $s(w) = -s(0y) = -s(y)$. Together with the previous paragraph, this implies that

$$(5) \quad i < 0 \Rightarrow K_{i,j} = K_{0,-1}\{0\}K_{-1-i,-j}.$$

We now choose five useful instances of (3), (4), and (5):

$$\begin{aligned} L &= K_{0,1} \cup K_{0,-1} \\ K_{0,1} &= \{\varepsilon\} \cup \{0\}K_{-1,1} \cup \{1\}K_{0,-1} \\ K_{0,-1} &= \{0\}K_{-1,-1} \cup \{1\}K_{0,1} \\ K_{-1,1} &= K_{0,-1}\{0\}K_{0,-1} \\ K_{-1,-1} &= K_{0,-1}\{0\}K_{0,1}. \end{aligned}$$

We thus have five equations in the five languages $L, K_{0,1}, K_{0,-1}, K_{-1,1}, K_{-1,-1}$. We associate these languages with nonterminals S, A, B, C, D , respectively. The system of set equations can now be interpreted as an instance of formula (1) for the grammar with the production rules

$$\begin{aligned} (6) \quad S &\rightarrow A \mid B \\ A &\rightarrow \varepsilon \mid 0C \mid 1B \\ B &\rightarrow 0D \mid 1A \\ C &\rightarrow B0B \\ D &\rightarrow B0A. \end{aligned}$$

According to the above derivation, the system of five languages is a solution of this grammar. In this case, A and S are the nullables, and we have $\text{ung}(V) = \{S\}$ and $\text{ung}^2(V) = \emptyset$. Therefore, by Theorem 4, grammar (6) has only one solution. This proves that L is the language generated by grammar (6).

5. The language of a pushdown automaton

In this section, we apply the method of Section 4 and the theory of Section 2 to determine a context-free grammar for the language accepted by a pushdown automaton. This corresponds to the classical theorem [Cho62] that the language accepted by a pushdown automaton is context-free. It is interesting, though, that the proof in [Cho62] is indirect and uses the fact that the intersection of a context-free language with a regular language is context-free. In modern courses, e.g., [HMU01, Sud06], the latter fact is always proved as an application of the theorem.

We use the kind of pushdown automata described in [Sud06]. A *pushdown automaton* is thus a sextuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Q_1)$, where Q is a finite state space, $q_0 \in Q$ is the start state, Σ is the input alphabet, Γ is the stack alphabet, $Q_1 \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma_+ \times \Gamma_+ \rightarrow \mathbb{P}(Q \times \Gamma_+)$ is the transition function. Here $\Sigma_+ = \Sigma \cup \{\varepsilon\}$ and $\Gamma_+ = \Gamma \cup \{\varepsilon\}$ are the alphabets Σ and Γ enriched with the symbol ε , which may be regarded as subsets of Σ^* and Γ^* . We use Q_1 instead of the conventional F because we want to use F for set valued functions as in Section 2.

In comparison with the pushdown automata of [HMu01], our automata thus lack a start symbol $Z_0 \in \Gamma$. In every step, they can only push a single symbol onto the stack, but they need not pop an element from the stack.

A *configuration* of the automaton is a triple $[q, w, \alpha]$ with $q \in Q$ (the current state), $w \in \Sigma^*$ (the remainder of the input), and $\alpha \in \Gamma^*$ (the current stack). The transitions between configurations are defined by

$$(7) \quad [q, v, \alpha] \vdash [r, w, \beta] \equiv \\ (\exists a \in \Sigma_+, X, Y \in \Gamma_+, \gamma \in \Gamma^* : \\ v = aw \wedge \alpha = X\gamma \wedge \beta = Y\gamma \wedge [r, Y] \in \delta(q, a, X)).$$

In words, in state q , if $[r, Y] \in \delta(q, a, X)$, a possible step is to read a from the input and X from the stack, to push Y onto the stack, and to proceed to state r , where each of a, X, Y can be ε (reading or pushing ε does nothing).

Computations of the automaton correspond to the reflexive transitive closure \vdash^* of \vdash . The language *accepted by the automaton* is defined by

$$(8) \quad L(M) = \{w \in \Sigma^* \mid \exists r \in Q_1 : [q_0, w, \varepsilon] \vdash^* [r, \varepsilon, \varepsilon]\}.$$

Acceptance thus relies on empty input, empty stack, and accepting state.

We now proceed to derive a context-free grammar for the language $L(M)$. For this purpose, we generalize the definition of $L(M)$ by allowing arbitrary initial and final states and initial stacks. For $q, r \in Q$ and $\alpha \in \Gamma^*$, we define

$$F(q, \alpha, r) = \{w \in \Sigma^* \mid [q, w, \alpha] \vdash^* [r, \varepsilon, \varepsilon]\}.$$

$F(q, \alpha, r)$ thus consists of the terminal strings that can be consumed by M together with the stack α by going from state q to state r .

Formula (8) implies

$$(9) \quad L(M) = \bigcup_{r \in Q_1} F(q_0, \varepsilon, r).$$

Just as in Section 4, we derive recurrence relations between the languages $F(q, \alpha, r)$. We first claim

$$(10) \quad F(q, \alpha, s)F(s, \beta, r) \subseteq F(q, \alpha\beta, r).$$

This is proved in

$$\begin{aligned} & u \in F(q, \alpha, s) \wedge v \in F(s, \beta, r) \\ \equiv & [q, u, \alpha] \vdash^* [s, \varepsilon, \varepsilon] \wedge [s, v, \beta] \vdash^* [r, \varepsilon, \varepsilon] \\ \Rightarrow & [q, uv, \alpha\beta] \vdash^* [s, v, \beta] \vdash^* [r, \varepsilon, \varepsilon] \\ \Rightarrow & uv \in F(q, \alpha\beta, r). \end{aligned}$$

As before, we restrict the attention to the languages $F(q, \alpha, r)$ with $\alpha \in \Gamma_+$, because the resulting grammar should only have a finite number of nonterminals.

Firstly, the case $\alpha = \varepsilon$. If $[q, w, \varepsilon] \vdash^* [r, \varepsilon, \varepsilon]$ uses one or more steps, the first transition necessarily uses a pair $[s, B] \in \delta(q, a, \varepsilon)$ for some $a \in \Sigma_+$ with $w = av$. This transition is then followed by a path $[s, v, B] \vdash^* [r, \varepsilon, \varepsilon]$. Otherwise, the path $[q, w, \varepsilon] \vdash^* [r, \varepsilon, \varepsilon]$ uses no steps, $q = r$ and $w = \varepsilon$. This is formalized in the linear proof:

$$\begin{aligned} & w \in F(q, \varepsilon, r) \\ \equiv & \{ \text{definition of } F \} \\ & [q, w, \varepsilon] \vdash^* [r, \varepsilon, \varepsilon] \\ \equiv & \{ \text{zero or more steps, let } a \in \Sigma_+ \} \\ & (q = r \wedge w = \varepsilon) \\ & \vee (\exists a, v, s, B : w = av \wedge [s, B] \in \delta(q, a, \varepsilon) \wedge [s, v, B] \vdash^* [r, \varepsilon, \varepsilon]) \\ \equiv & \{ \text{definition of } F \} \\ & (q = r \wedge w = \varepsilon) \vee (\exists a, s, B : [s, B] \in \delta(q, a, \varepsilon) \wedge w \in aF(s, B, r)). \end{aligned}$$

This implies that $F(q, \varepsilon, r)$ is the union of all languages $aF(s, B, r)$ where $a \in \Sigma_+$ and $[s, B] \in \delta(q, a, \varepsilon)$ can be chosen arbitrarily, unified with ε if $q = r$. In a formula:

$$(11) \quad F(q, \varepsilon, r) = \{\varepsilon \mid q = r\} \cup \bigcup \{aF(s, B, r) \mid a \in \Sigma_+, [s, B] \in \delta(q, a, \varepsilon)\}.$$

For the case where $\alpha = A$ for a single nonterminal $A \in V$, we need to distinguish two kinds of steps between configurations: a \vdash_1 step takes one element from the stack, a \vdash_0 step does not. We thus partition relation \vdash into subrelations \vdash_0 and \vdash_1 defined by

$$\begin{aligned} [q, v, \alpha] \vdash_0 [r, w, \beta] &\equiv \text{formula (7) with } X = \varepsilon, \\ [q, v, \alpha] \vdash_1 [r, w, \beta] &\equiv \text{formula (7) with } X \in \Gamma. \end{aligned}$$

We use these relations to define

$$\begin{aligned} F'(q, \alpha, r) &= \{w \in \Sigma^* \mid [q, w, \alpha] \vdash_1^* [r, \varepsilon, \varepsilon]\}, \\ F''(q, \alpha, r) &= \{w \in \Sigma^* \mid [q, w, \alpha] \vdash_0^* [r, \varepsilon, \varepsilon]\}, \end{aligned}$$

where \vdash_1^* and \vdash_0^* are the relational compositions. Since every nonempty path starts either with a \vdash_1 step or with a \vdash_0 step, we have

$$(12) \quad F(q, \alpha, r) = F'(q, \alpha, r) \cup F''(q, \alpha, r).$$

For $F'(q, A, r)$, we can unfold the first \vdash_1 step in the same way as in the computation of (11), because the \vdash_1 step does not increase the stack size. We thus obtain

$$F'(q, A, r) = \bigcup \{aF(s, B, r) \mid a \in \Sigma_+, [s, B] \in \delta(q, a, A)\}.$$

On the other hand, we have

$$F''(q, A, r) \subseteq \bigcup_s F(q, \varepsilon, s)F(s, A, r),$$

by choosing $s := q$, because $\varepsilon \in F(q, \varepsilon, q)$ by (11) and $F''(q, A, r) \subseteq F(q, A, r)$ by (12) (this argument was given by a referee; our original argument was clumsy).

We use formula (10) to obtain converse inclusions. In this way, we obtain for all $A \in \Gamma$ the equality:

$$(13) \quad F(q, A, r) = \bigcup \{aF(s, B, r) \mid a \in \Sigma_+, [s, B] \in \delta(q, a, A)\} \\ \cup \bigcup \{F(q, \varepsilon, s)F(s, A, r) \mid s \in Q\}.$$

The recurrence equations (9), (11), (13) are transformed into production rules for a context-free grammar. We take S as starting symbol, and the nonterminals $\langle q, A, r \rangle$ for all $q, r \in Q$ and $A \in \Gamma_+$. In view of formula (9), we take the production rules

$$(14) \quad S \rightarrow \langle q_0, \varepsilon, r \rangle \text{ for all } r \in Q_1.$$

In view of formula (11) we take the rules

$$(15) \quad \langle q, \varepsilon, q \rangle \rightarrow \varepsilon \text{ for all } q \in Q.$$

In view of (11) and (13) we take the rules

$$(16) \quad \langle q, A, r \rangle \rightarrow a\langle s, B, r \rangle \text{ whenever } [s, B] \in \delta(q, a, A) \text{ for } A \in \Gamma_+.$$

In view of formula (13) we further take the rules

$$(17) \quad \langle q, A, r \rangle \rightarrow \langle q, \varepsilon, s \rangle \langle s, A, r \rangle \text{ for all } A \in \Gamma \text{ and } q, r, s \in Q.$$

Let G be the grammar defined by the four families of production rules (14), (15), (16), (17). Let function F be defined by $F(S) = L(M)$ and $F(\langle q, A, r \rangle) = F(q, A, r)$. It follows from (9), (11), (13) that function F is a solution of grammar G .

Theorem 6 *The language $L(M)$ is the language generated by grammar G .*

Proof. By Theorem 2, it suffices to prove that function F constructed above has a filtration. For this purpose, we take the filtration according to computation length defined by

$$\begin{aligned} F_n(S) &= \{w \mid \exists r \in Q_1, k < n - 1 : [q_0, w, \varepsilon] \vdash^k [r, \varepsilon, \varepsilon]\}, \\ F_n(q, A, r) &= \{w \mid \exists k < n : [q, w, A] \vdash^k [r, \varepsilon, \varepsilon]\}, \end{aligned}$$

where $F_n(q, A, r)$ is an abbreviation of $F_n(\langle q, A, r \rangle)$.

It is clear that $F(X) = \bigcup F_n(X)$ and that $F_0(X) = \emptyset$ for all nonterminals X . For the remainder of the proof that $(F_n \mid n)$ is a filtration of F , we need to show:

$$\begin{aligned} F_{n+1}(S) &\subseteq \bigcup_{r \in Q_1} F_n(q_0, \varepsilon, r), \\ F_{n+1}(q, \varepsilon, r) &\subseteq \{\varepsilon \mid q = r\} \cup \bigcup \{aF_n(s, B, r) \mid a \in \Sigma_+, [s, B] \in \delta(q, a, \varepsilon)\}, \\ F_{n+1}(q, A, r) &\subseteq \bigcup \{aF_n(s, B, r) \mid a \in \Sigma_+, [s, B] \in \delta(q, a, A)\} \\ &\quad \cup \bigcup \{F_n(q, \varepsilon, s)F_n(s, A, r) \mid s \in Q\}. \end{aligned}$$

The first inclusion is trivial. The proofs of the other two inclusions are completely analogous to the proofs of (11) and (13). \square

Remarks. If the pushdown automaton has k states and g stack symbols, the resulting grammar has $1 + k^2(g + 1)$ nonterminals. The number of production rules is $f + k(1 + D + k^2g)$ when f is the number of accepting states and D is the number of transitions. In practical cases, these numbers are uncomfortably large. The grammar of [Sud06, Section 7.3] uses the same nonterminals but considerably more production rules. The grammar of [HMu01, Section 6.3.2] needs even more production rules but they accommodate more complex pushdown automata.

Theorem 4 is not applicable in this case: because of the rules (17) and (15), the set of the nonterminals $\langle q, A, r \rangle$ with $A \in \Gamma$ is unguarded. Also rule (16) can introduce unguardedness when $a = \varepsilon$.

One may wonder whether the grammar can be transformed in such a way that it has no nonempty unguarded set anymore, preferably without increasing the number of production rules.

6. Conclusion

When a language is not given by how its strings are generated, but by some other means, it is not directly clear how to derive a grammar for it. In this paper, we propose a method for this, which consists of three steps: first to derive equations for the language, then to recognize in this set of equations an instance of formula (1) for a specific grammar, and finally to argue that the language equals the smallest solution, which is the language generated by the grammar (Theorem 2).

In Sections 4 and 5, we applied this method fruitfully to two nontrivial cases. The third step of the method can be done by two different arguments: Theorem 4 gives a sufficient condition that a grammar has a unique solution. This was used for the example of Section 4. In Section 5, we used Theorem 2, which gives a sufficient condition that a solution of a grammar equals the language generated by the grammar.

Of course, the method cannot be always successful. If the language is not context-free, the method must fail. If the language is known to be context-free, there may be easier ways to get a grammar for it. The method aims therefore primarily at languages that are suspected but not known to be context-free.

This paper was directly inspired by teaching courses on automata and formal languages, first in 1987 in Austin, Texas, and the last years in Groningen. These courses are difficult for many students and, in my experience, do not combine well with the teaching of equational reasoning. Correctness of **while** programs is better suited for that purpose. Yet, as this paper shows, there are cases, where it is useful or even necessary to regard a context-free grammar as a system of equations in languages, and Section 5 shows that, for this purpose, equational reasoning can be essential.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

- [Bod65a] Bodnarchuk VG (1965) The metrical space of events, part I. *Cybernet Syst Anal* 1(1):20–24
- [Bod65b] Bodnarchuk VG (1965) The metrical space of events, part II. *Cybernet Syst Anal* 1(4):27–36
- [Cho62] Chomsky N (1962) Context-free grammar and pushdown storage. Quarterly Progress Report, MIT Research Lab. in Electronics 65:187–194
- [HMu01] Hopcroft JE, Motwani R, Ullman JD (2001) Introduction to automata theory, languages and computation. Addison-Wesley, Reading
- [Kup97] Kupka I (1997) Unique fixpoints in complete lattices with applications to formal languages and semantics. In: Freksa C, Jantzen M, Valk R (eds) Foundations of Computer Science: potential-theory-cognition, vol 1337 of LNCS. Springer, New York, pp 107–115
- [McW71] McWhirter IP (1971) Substitution expressions. *J Comput Syst Sci* 5(6):629–637
- [MG87] Mandrioli D, Ghezzi C (1987) Theoretical foundations of Computer Science. Wiley, New York
- [Red65] Red'ko VN (1965) Some aspects of the theory of languages. *Cybernet Syst Anal* 1(4):15–26
- [Sud06] Sudkamp TA (2006) Languages and machines, 3rd edn. Pearson, Boston

Received 31 March 2009

Accepted in revised form 27 August 2009 by E.C.R. Hehner

Published online 2 October 2009